Contents

Network Commands **(ADMN)**

tcp	TCP start/stop script
telnetd	DARPA TELNET protocol server
tftpd	DARPA Trivial File Transfer Protocol server
timed	time server daemon
timedc	timed control program
trace	routing tools
trpt	transliterate protocol trace

intro

introduction to network maintenance and operation commands

Description

This section contains information related to network operation and maintenance. It describes a variety of commands: *slink*, to bring up the transport; *ifconfig*, and *slattach*, to configure network interfaces; *ping*, to test status of remote hosts; *trpt*, to display packet-tracing information; to invoke network services; and and other network administration functions.

arp

address resolution display and control

Syntax

arp hostname
arp -a [namelist] [corefile]
arp -d hostname
arp -s hostname ether_addr **[temp] [pub] [trail]**
arp -f filename

Description

The *arp* program displays and modifies the Internet-to-Ethernet address-translation table, which is normally maintained by the address-resolution protocol (*arp*(ADMP)).

When *hostname* is the only argument, *arp* displays the current ARP entry for *hostname*. The host may be specified by name or number, using Internet dot notation. [See *hosts*(ADMN) and *inet*(ADMP).]

Options are interpreted as follows:

-a [*namelist*] [*corefile*]
Display all of the current ARP entries by reading the table from the file **corefile** (default **/dev/kmem**) based on the kernel file **namelist** (default **/unix**).

-d Delete an entry for the host whose name is *hostname*. (This can be performed only by the super user.)

-s *hostname ether_addr* **[temp] [pub] [trail]**
Create an ARP entry for the host whose name is *hostname* with the Ethernet address *ether_addr*. The Ethernet address is given as six colon-separated, two-digit hexadecimal numbers. The entry will be permanent unless the argument **temp** is specified on the command line. If **pub** is specified, the entry will be published: that is, this system will act as an ARP server, responding to requests for *hostname* even though the host address is not an address of the local host. If **trail** is specified, trailer encapsulations are to be used with this host. *N.B.* Trailers are a link-dependent issue. Currently, no known LLI-compliant ethernet driver suppports trailers, and it is unwise to advertise them, unless it is certain that the link layer can handle them.

-f *filename*
Read the file **filename** and set multiple entries in the ARP tables. Entries in the file should be of the form:

hostname ether_addr [**temp**] [**pub**] [**trail**]

with argument meanings as given above.

See Also

inet(SLIB), arp(ADMP), ifconfig(ADMN).

drvconf

configure TCP/IP ethernet drivers

Syntax

/etc/drvconf

Description

/etc/drvconf is used to configure TCP/IP to use a particular ethernet driver. It prompts with a list of possible drivers and asks the user to select one. The TCP/IP configuration files */etc/strcf* and */etc/tcp* are then modified to use the appropriate driver. The driver must be installed on the system when *drvconf* is run.

See Also

strcf(SFF), tcp(ADMN), idmknod(ADMN).

Bugs

As distributed, this command only supports drivers for the AT/386.

fingerd

remote user information server

Syntax

/etc/fingerd

Description

fingerd is a server that provides a network interface to the *finger*(TC) program (or, on some other systems, the *name* program). This interface allows *finger* to display information about remote users.

fingerd listens for TCP connections on the *finger* port. (See *services(SFF).)* For each connection, *fingerd* reads a single input line (terminated by a <CRLF>), passes the line to *finger*, and copies the output of *finger* to the user on the client machine.

fingerd is started by the super-server *inetd*, and therefore must have an entry in *inetd*'s configuration file **/etc/inetd.conf.** [See *inetd*(ADMN) and *inetd.conf*(SFF).]

For it to work, *fingerd* needs to have a **/usr/local/bin** directory created and then linked to **/usr/bin/finger**.

See Also

finger(TC), inetd(ADMN), inetd.conf(SFF), services(SFF), RFC 742.

Warning

Connecting to *fingerd* using TELNET (see *telnet*(TC)) can have unpredictable consequences and is not recommended.

ftpd

DARPA Internet File Transfer Protocol server

Syntax

/etc/ftpd [**-d**] [**-l**] [**-t***timeout*]

Description

ftpd is the DARPA Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the ftp service specification; see *services(SFF).*

ftpd is started by the super-server *inetd*, and therefore must have an entry in *inetd*'s configuration file **/etc/inetd.conf.** [See *inetd(ADMN)* and *inetd.conf(SFF).]*

If the **-d** option is specified, debugging information is written to the syslog.

If the **-l** option is specified, each FTP session is logged in the syslog.

The FTP server will timeout an inactive session after 15 minutes. If the **-t** option is specified, the inactivity timeout period will be set to *timeout*.

The FTP server currently supports the following FTP requests; case is not distinguished.

Request	**Description**
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (ls -l)
MKD	make a directory
MODE	specify data transfer *mode*
NLST	give name list of files in directory (ls)
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session

RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name
STOR	store a file
STOU	store a file with a unique name
STRU	specify data transfer *structure*
TYPE	specify data transfer *type*
USER	specify user name
XCUP	change to parent of current working directory
XCWD	change working directory
XMKD	make a directory
XPWD	print the current working directory
XRMD	remove a directory

The remaining FTP requests specified in Internet RFC 959 are recognized, but not implemented.

The FTP server will abort an active file transfer only when the ABOR command is preceded by a Telnet Interrupt Process (IP) signal and a Telnet Synch signal in the command Telnet stream, as described in Internet RFC 959.

ftpd interprets file names according to the globbing conventions used by *sh*(C). This allows users to utilize the metacharacters *?[]{}˜.

ftpd authenticates users according to three rules.

1) The user name must be in the password data base **/etc/passwd** and not have a null password. In this case, a password must be provided by the client before any file operations can be performed.

2) The user name must not appear in the file **/etc/ftpusers**.

3) If the user name is anonymous or ftp, an anonymous ftp account must be present in the password file (user ftp). In this case, the user is allowed to log in by specifying any password. (By convention, this is given as the client host's name.)

In the last case, *ftpd* takes special measures to restrict the client's access privileges. The server performs a *chroot*(2) command to the home directory of the ftp user. To make sure system security is not breached, it is recommended that the ftp subtree be constructed with care; the following rules are recommended. (Note: ˜ftp means "the home directory of user ftp")

˜ftp)
: Make it so the home directory is owned by ftp and unwritable by anyone.

~ftp/bin)
: Make it so this directory is owned by the superuser and unwritable by anyone. The program *ls*(C) must be present to support the list commands. This program should have mode 111.

~ftp/etc)
: Make it so this directory owned by the superuser and unwritable by anyone. The files *passwd*(SFF) and *group*(SFF) must be present for the *ls* command to work properly. These files should be mode 444.

~ftp/pub)
: Make this directory mode 777 and owned by ftp. Users should then place files that are to be accessible via the anonymous account in this directory.

See Also

ftp(TC), syslog(SLIB)

Notes

The anonymous account is inherently dangerous and should avoided when possible.

The server must run as the superuser to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the superuser only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

Files

/etc/ftpusers - restricted user list
/etc/passwd - the user database
/etc/group - the group database
/usr/adm/syslog - the system log file

The following files are needed for anonymous ftp:

~ftp/etc/passwd - used by ~ftp/bin/ls
~ftp/etc/group - used by ~ftp/bin/ls
~ftp/bin/ls - to support the LIST and NLST commands

In addition, if your /bin/ls is linked with shared libraries, you will need to copy /shlib/libc_s to ~ftp/shlib/libc_s. If your implementation is using the SIOCSOCKSYS ioctl, you will need to run the *mknod*(ADMN) command on ~ftp/dev/socksys.

mknoD

hostname

host name resolution description

Description

Hostnames are domains, where a domain is a hierarchical, dot-separated list of subdomains; for example, the machine laiter, in the Lachman subdomain of the COM subdomain of the ARPANET would be represented as

laiter.Lachman.COM

(with no trailing dot).

Hostnames are often used with network client and server programs, which must generally translate the name to an address for use. (This function is generally performed by the library routine *gethostbyname* (SSC).) Hostnames are resolved by the internet name resolver in the following fashion.

If the name consists of a single component, i.e. contains no dot, and if the environment variable "HOSTALIASES" is set to the name of a file, that file is searched for an string matching the input hostname. The file should consist of lines made up of two white-space separated strings, the first of which is the hostname alias, and the second of which is the complete hostname to be substituted for that alias. If a case-sensitive match is found between the hostname to be resolved and the first field of a line in the file, the substituted name is looked up with no further processing.

If the input name ends with a trailing dot, the trailing dot is removed, and the remaining name is looked up with no further processing.

If the input name does not end with a trailing dot, it is looked up in the local domain and its parent domains until either a match is found or fewer than 2 components of the local domain remain. For example, in the domain CHI.Lachman.COM, the name flaime.STG will be checked first as flaime.STG.CHI.Lachman.COM and then as flaime.STG.Lachman.COM. Flaime.STG.COM will not be tried, as the there is only one component remaining from the local domain.

See Also

gethostent(SFF),resolver(ADMN), mailaddr(ADMN), named(ADMN).
RFC883.

ifconfig

configure network interface parameters

Syntax

/etc/ifconfig interface address_family [address [dest_address]]
  [parameters]

/etc/ifconfig interface [protocol_family]

Description

ifconfig is used to assign an address to a network interface and/or configure network interface parameters; it defines the network address of each interface present on a machine. *ifconfig* is run at system start-up time via *tcp(1M)*. *ifconfig* may be run at other times to redefine an interface's address or other operating parameters. (For example, *slattach*(ADMN) also runs *ifconfig*.)

The interface parameter is a string of the form "name unit", for example, "en0".

Since an interface may receive transmissions in differing protocols, each of which may require a separate naming scheme, it is necessary to specify the address_family, which may change the interpretation of the remaining parameters. Currently, only the Internet address family is supported: thus, the only valid value for address_family is inet.

For the DARPA-Internet family, the address is either a host name or a DARPA Internet address expressed in the Internet standard "dot notation". (Host name translation is performed either by the name server or by an entry in **/etc/hosts.** [See *named*(ADMN) and *hosts*(ADMN).] Internet "dot notation" is described in *hosts*(ADMN) and *inet*(ADMP). Other address families may use different notations.)

The following parameters may be set with *ifconfig*:

up	Mark an interface "up". This may be used to enable an interface after an "ifconfig down". It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down	Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.
detach	Remove an interface from the system. This command is applicable to transient interfaces only, such as serial line interfaces.
trailers	Request the use of a trailer link level encapsulation when sending (default). If a network interface supports trailers, the system will, when possible, encapsulate outgoing messages in a manner that minimizes the number of memory-to-memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see *arp*(ADMP); currently, only 10 Mb/s Ethernet), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. This is currently used by Internet protocols only.
-trailers	Disable the use of a trailer-link-level encapsulation.
arp	Enable the use of the Address Resolution Protocol in mapping between network level addresses and link-level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10Mb/s Ethernet addresses. This option is not applicable in the STREAMS environment. Use of arp for an interface is specified in **/etc/strcf**. The arp driver will be opened when the STREAMS stack is built.
-arp	Disable the use of the Address Resolution Protocol.
metric *n*	Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol. Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.
debug	Enable driver-dependent debugging code; usually, this turns on extra console error logging.
-debug	Disable driver-dependent debugging code.

netmask *mask* (Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table *networks(SFF)*. The mask contains 1's for the bit positions in the 32-bit address, which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

dstaddr Specify the address of the correspondent on the other end of a point-to-point link.

broadcast (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

onepacket Enable the *one-packet* mode of operation (used for interfaces that cannot handle back-to-back packets) The keyword **onepacket** must be followed by two numeric parameters, giving the small packet size and threshold, respectively. If small packet detection is not desired, these values should be zero. See *tcp*(ADMP) for an explanation on one-packet mode.

-onepacket Disable one-packet mode.

ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, *ifconfig* will report only the details specific to that protocol family.

Only the superuser may modify the configuration of a network interface.

Diagnostics

Messages indicating the specified interface does not exit, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

Files

/etc/slattach
calls *ifconfig* to start serial lines

See Also

arp(ADMN), tcp(ADMN), netstat(TC), hosts(SFF), networks(SFF), strcf(ADMN), arp(ADMP), tcp(ADMP).

inetd

internet super server

Syntax

/etc/inetd [**-d**] [configuration file]

Description

inetd listens on multiple ports for incoming connection requests. When it receives a request, it spawns the appropriate server. The use of a superserver allows other servers to be spawned only when needed and to terminate when they have satisfied a particular request.

The mechanism is as follows: *inetd* is started by the superuser (usually during init 2, if **/etc/tcp** is linked to **/etc/rc2.d/S*nn*tcp**.). To obtain information about the servers it needs to spawn, *inetd* reads its configuration file (by default, */etc/inetd.conf* (SFF)) and issues a call to *getservbyname* . [See *getservent* (SLIB).] (Note that **/etc/services** and **/etc/protocols** must be properly configured.) *inetd* then creates a socket for each server and binds each socket to the port for that server. It does a *listen* (SSC) on all connection-based sockets (that is, stream rather than datagram), and waits, using *select* (SSC), for a connection or datagram.

- When a connection request is received on a listening (stream) socket, *inetd* does an *accept* (SSC), thereby creating a new socket. (*inetd* continues to listen on the original socket for new requests). *inetd* forks, dups, and execs the appropriate server, passing it any server program arguments specified in *inetd* 's configuration file. The invoked server has I/O to **stdin**, **stdout**, and **stderr** done to the new socket; this connects the server to the client process. (Some built-in, internal services are performed via function calls rather than child processes.)

- When there is data waiting on a datagram socket, *inetd* forks, dups, and execs the appropriate server, passing it any server program arguments; unlike a connection-based server, a datagram server has I/O to **stdin**, **stdout**, and **stderr** done to the original socket. If the datagram socket is marked as *wait* (which corresponds to an entry in *inetd* 's configuration file), the invoked server must process the message before *inetd* considers the socket available for new connections. If the datagram socket is marked as *nowait, inetd* continues to process incoming messages on that port. *tftpd* is an exceptional case: although its entry in *inetd* 's configuration file must be *wait* (to avoid contention for the port), *inetd* is able to continue processing new messages on the port.

The following servers may be started by *inetd*: *fingerd*, *ftpd*, *rexecd*, *rlogind*, *rshd*, *telnetd*, and *tftpd*. *inet* must also start several internal services: these are described in *inetd.conf*(SFF). Do **not** arrange for *inetd* to start *rwhod*, or any NFS server.

inetd rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread.

The **-d** option turns on socket-level debugging and prints debugging information to **stdout**.

Files

/etc/inetd.conf
/etc/protocols
/etc/services

See Also

fingerd(ADMN), ftpd(ADMN), rexecd(ADMN), rlogind(ADMN), rshd(ADMN), telnetd(ADMN), tftpd(ADMN), inetd.conf(SFF), protocols(SFF), services(SFF).

ldsocket

load socket configuration

Syntax

ldsocket [**-v**] [**-c** file]

Description

ldsocket initializes the System V STREAMS TCP/IP Berkeley networking compatability interface, which is an alternate stream head supporting the *socket*(SSC) system call family. *ldsocket* loads the kernel with associations between the protocol family, type and number triplets passed to the *socket* system call, and the STREAMS devices supporting those protocols. *ldsocket* reads the file **/etc/sockcf** to obtain configuration information, and must be run before the Berkeley networking interface can be used.

The following options may be specified on the *ldsocket* command line:

-c *file*	Use *file* instead of **/etc/sockcf**.
-v	Use verbose mode (in which a message is written to **stderr** for each protocol loaded).

Files

/etc/sockcf

See Also

sockcf(SFF), intro(ADMP), socket(SSC).

lmail

handle local mail delivery from sendmail

Syntax

lmail user ...

Description

lmail interprets incoming mail received from *sendmail*(ADMN), and delivers it to the specified user on the local machine. It locks the user's mailbox using the *mail*(TC) locking mechanism.

See Also

mail(TC), sendmail(ADMN).

mailaddr

mail addressing description

Description

Mail addresses are based on the ARPANET protocol listed at the end of this manual page. These addresses are in the general format

user@domain

where a domain is a hierarchical dot separated list of subdomains. For example, the address

stevea@laiter.lachman.com

is normally interpreted from right to left: the message should go to the Lachman gateway, after which it should go to the local host laiter. When the message reaches laiter it is delivered to the user "stevea".

Unlike some other forms of addressing, this does not imply any routing. Thus, although this address is specified as an RFC822 address, it might travel by an alternate route if that were more convenient or efficient. For example, at Lachman, the associated message would probably go directly to laiter over the Ethernet rather than going via the Lachman mail gateway.

Abbreviation.

Under certain circumstances it may not be necessary to type the entire domain name. In general, anything following the first dot may be omitted if it is the same as the domain from which you are sending the message. For example, a user on "laisagna.Lachman.COM'' could send to "stevea@laiter" without adding the "Lachman.COM" since it is the same on both sending and receiving hosts.

Certain other abbreviations may be permitted as special cases. For example, at Lachman, Internet hosts may be referenced without adding the "Lachman.COM" as long as their names do not conflict with a local host name.

Compatibility.

Certain old address formats are converted to the new format to provide compatibility with the previous mail system. In particular,

user@host.ARPA

is allowed and

```
            host:user
```

is converted to

```
            user@host
```

to be consistent with the *rcp*(1) command.

Also, the syntax

```
            host!user
```

is converted to:

```
            user@host.UUCP
```

This is normally converted back to the “host!user” form before being sent on for compatibility with older UUCP hosts.

The current implementation is not able to route messages automatically through the UUCP network. Until that time you must explicitly tell the mail system which hosts to send your message through to get to your final destination.

Case Distinctions.

Domain names (i.e., anything after the “@” sign) may be given in any mixture of upper and lower case with the exception of UUCP hostnames. Most hosts accept any combination of case in user names, with the notable exception of MULTICS sites.

Route-addrs.

Under some circumstances it may be necessary to route a message through several hosts to get it to the final destination. Normally this routing is done automatically, but sometimes it is desirable to route the message manually. Addresses which show these relays are termed “route-addrs.” These use the syntax:

```
            <@hosta,@hostb:user@hostc>
```

This specifies that the message should be sent to hosta, from there to hostb, and finally to hostc. This path is forced even if there is a more efficient path to hostc.

Route-addrs occur frequently on return addresses, since these are generally augmented by the software at each host. It is generally possible to ignore all but the “user@domain” part of the address to determine the actual sender.

Postmaster.

Every site is required to have a user or user alias designated "postmaster" to which problems with the mail system may be addressed.

Other Networks.

Some other networks can be reached by giving the name of the network as the last component of the domain. *This is not a standard feature* and may not be supported at all sites. For example, messages to CSNET or BITNET sites can often be sent to "user@host.CSNET" or "user@host.BITNET" respectively.

Bugs

The RFC822 group syntax ("group:user1,user2,user3;") is not supported except in the special case of "group:;" because of a conflict with old berknet-style addresses.

Route-Address syntax is ugly.

UUCP- and RFC822-style addresses do not coexist politely.

See Also

mailx(TC), sendmail(ADMN). RFC822.

mconnect

connect to SMTP mail server socket

Syntax

mconnect [**-p** *port*] [**-r**] [*hostname*]

Description

Mconnect opens a connection to the mail server on a given host, so that it can be tested independently of all other mail software. If no host is given, the connection is made to the local host. Servers expect to speak the Simple Mail Transfer Protocol (SMTP) on this connection. Exit by typing the "quit" command. Typing end-of-file will cause an end of file to be sent to the server. An interrupt closes the connection immediately and exits.

Options

-p Specify the port number instead of the default SMTP port (number 25) as the next argument.

-r "Raw" mode: disable the default line buffering and input handling. This gives you a similar effect as *telnet* to port number 25, not very useful.

Files

/usr/lib/sendmail.hf	Help file for SMTP commands

See Also

sendmail(ADMN).
RFC821.

mkhosts

make node name commands

Syntax

/etc/mkhosts

Description

mkhosts makes the simplified forms of the *rcmd*(TC) and *rlogin*(TC) commands. For each node listed in **/etc/hosts**, *mkhosts* creates a link to **/usr/bin/rcmd** in **/usr/hosts**. Each link's name is the same as the node's official name in **/etc/hosts.**

See Also

rcmd (TC), rlogin(TC).

named

Internet domain name server

Syntax

named [**-d** *debuglevel*] [**-p** *port#*] [**-b** *bootfile*]

Description

named is the Internet domain name server. (See RFC1035 for more details on the Internet name-domain system.) Without any arguments, *named* will read the default boot file **/etc/named.boot**, read any initial data, and listen for queries.

Options are:

-d Print debugging information. A number after the **d** determines the level of messages printed.

-p Use a different port number. The default is the standard port number as listed in **/etc/services**.

-b Use an alternate boot file. This is optional and allows you to specify a file with a leading dash.

Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data. If multiple boot files are specified, only the last is used. Lines in the boot file cannot be continued on subsequent lines. The following is a small example:

```
;
;
;          boot file for name server
;
directory  /usr/local/lib/named
; type     domain                source host/file             backup file

cache      .                                                  root.cache
primary    Berkeley.EDU          berkeley.edu.zone
primary    32.128.IN-ADDR.ARPA   ucbhosts.rev
secondary  CC.Berkeley.EDU       128.32.137.8 128.32.137.3   cc.zone.bak
secondary  6.32.128.IN-ADDR.ARPA 128.32.137.8 128.32.137.3   cc.rev.bak
primary    0.0.127.IN-ADDR.ARPA                               localhost.rev
forwarders 10.0.0.78 10.2.0.78
; slave
```

The "directory" line causes the server to change its working directory to the directory specified. This can be important for the correct processing of $INCLUDE files in primary zone files.

The "cache" line specifies that data in "root.cache" is to be placed in the backup cache. Its main use is to specify data such as locations of root domain servers. This cache is not used during normal operation, but is used as "hints" to find the current root servers. The file "root.cache" is in the same format as "berkeley.edu.zone". There can be more than one "cache" file specified. The cache files are processed in such a way as to preserve the time-to-live's of data dumped out. Data for the root nameservers is kept artificially valid if necessary.

The first "primary" line states that the file "berkeley.edu.zone" contains authoritative data for the "Berkeley.EDU" zone. The file "berkeley.edu.zone" contains data in the master file format described in RFC1035. All domain names are relative to the origin, in this case, "Berkeley.EDU" (see below for a more detailed description). The second "primary" line states that the file "ucbhosts.rev" contains authoritative data for the domain "32.128.IN-ADDR.ARPA," which is used to translate addresses in network 128.32 to hostnames. Each master file should begin with an SOA record for the zone (see below).

The first "secondary" line specifies that all authoritative data under "CC.Berkeley.EDU" is to be transferred from the name server at 128.32.137.8. If the transfer fails it will try 128.32.137.3 and continue trying the addresses, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain. The first non-dotted-quad address on this line will be taken as a filename in which to backup the transferred zone. The name server will load the zone from this backup file if it exists when it boots, providing a complete copy even if the master servers are unreachable. Whenever a new copy of the domain is received by automatic zone transfer from one of the master servers, this file will be updated. The second "secondary" line states that the address-to-hostname mapping for the subnet 128.32.136 should be obtained from the same list of master servers as the previous zone.

The "forwarders" line specifies the addresses of sitewide servers that will accept recursive queries from other servers. If the boot file specifies one or more forwarders, then the server will send all queries for data not in the cache to the forwarders first. Each forwarder will be asked in turn until an answer is returned or the list is exhausted. If no answer is forthcoming from a forwarder, the server will continue as it would have without the forwarders line unless it is in "slave" mode. The forwarding facility is useful to cause a large sitewide cache to be generated on a master, and to reduce traffic over links to outside servers. It can also be used to allow servers to run that do not have access directly to the Internet, but wish to act as though they do.

The "slave" line (shown commented out) is used to put the server in slave mode. In this mode, the server will only make queries to forwarders. This option is normally used on machine that wish to run a server but for physical or administrative reasons cannot be given access to the Internet, but have access to a host that does have access.

The "sortlist" line can be used to indicate networks that are to be preferred over other, unlisted networks. Queries for host addresses from hosts on the same network as the server will receive responses with local network addresses listed first, then addresses on the sort list, then other addresses. This line is only acted on at initial startup. When reloading the nameserver with a SIGHUP, this line will be ignored.

The master file consists of control information and a list of resource records for objects in the zone of the forms:

```
$INCLUDE <filename> <opt_domain>
$ORIGIN <domain>
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>
```

where *domain* is ". " for root, "@" for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with ".", the current origin is appended to the domain. Domain names ending with "." are unmodified. *opt_domain* field is used to define an origin for the data in an included file. It is equivalent to placing a $ORIGIN statement before the first line of the included file. The field is optional. Neither the *opt_domain* field nor $ORIGIN statements in the included file modify the current origin for this file. The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero, meaning the minimum value specified in the SOA record for the zone. The *opt_class* field is the object address type; currently only one type is supported, **IN**, for objects connected to the DARPA Internet. The *type* field contains one of the following tokens; the data expected in the *resource_record_data* field is in parentheses.

A	a host address (dotted quad)
NS	an authoritative name server (domain)
CNAME	the canonical name for an alias (domain)
SOA	marks the start of a zone of authority (domain of originating host, domain address of maintainer, a serial number and the following parameters in seconds: refresh, retry, expire and minimum TTL (see RFC1035))
MB	a mailbox domain name (domain)
MG	a mail group member (domain)

MR	a mail rename domain name (domain)
MX	a mail exchange record
NULL	a null resource record (no format or data)
WKS	a well-known service description (not implemented yet)
PTR	a domain name pointer (domain)
HINFO	host information (cpu_type OS_type)
MINFO	mailbox or mail list information (request_domain error_domain)

Resource records normally end at the end of a line, but may be continued across lines between opening and closing parentheses. Comments are introduced by semicolons and continue to the end of the line.

Each master zone file should begin with an SOA record for the zone. An example SOA record is as follows:

```
@     IN    SOA  ucbvax.Berkeley.EDU.  rwh.ucbvax.Berkeley.EDU. (
                      2.89   ; serial
                      10800; refresh
                      3600   ; retry
                      3600000     ; expire
                      86400 )     ; minimum
```

The SOA lists a serial number, which should be changed each time the master file is changed. Secondary servers check the serial number at intervals specified by the refresh time in seconds; if the serial number changes, a zone transfer will be done to load the new data. If a master server cannot be contacted when a refresh is due, the retry time specifies the interval at which refreshes should be attempted until successful. If a master server cannot be contacted within the interval given by the expire time, all data from the zone is discarded by secondary servers. The minimum value is the time-to-live used by records in the file with no explicit time-to-live value.

Notes

The boot file directives "domain" and "suffixes" have been obsoleted by a more useful resolver based implementation of suffixing for partially qualified domain names. The prior mechanisms could fail under a number of situations, especially when then local nameserver did not have complete information.

The following signals have the specified effect when sent to the server process using the *kill* (C) command.

SIGHUP	Causes server to read named.boot and reload database.
SIGINT	Dumps current data base and cache to **/usr/tmp/named_dump.db**.
SIGIOT	Dumps statistics data into /usr/tmp/named.stats if the server is compiled -DSTATS. Statistics data is appended to the file.
SIGSYS	Dumps the profiling data in /usr/tmp if the server is compiled with profiling (server forks, chdirs and exits).
SIGTERM	Dumps the primary and secondary database files. Used to save modified data on shutdown if the server is compiled with dynamic updating enabled.
SIGUSR1	Turns on debugging; each SIGUSR1 increments debug level.
SIGUSR2	Turns off debugging completely.

Files

/etc/named.boot	name server configuration boot file
/etc/named.pid	the process id
/usr/tmp/named.run	debug output
/usr/tmp/named_dump.db	dump of the name servers database
/usr/tmp/named.stats	nameserver statistics data

See Also

kill(C), gethostent(SLIB), signal(S), sigset(S), resolver(SFF), resolver(ADMN), hostname(ADMP).
RFC974, RFC1034, RFC1035, *Name Server Operations Guide for BIND*.

netlogin

network login program

Syntax

netlogin [**-p**] [**-r** remotehost] [name] [env-var]

Description

Netlogin is a derivative of the *login*(TC) command. It provides facilities that *telnetd*(ADMN) and *rlogind(ADMN)* need, such as preserving the environment, and support for automatically logging users in. *Netlogin* takes the following options:

-p Preserve the environment. This is used by *telnetd* to pass information obtained via terminal type negotiation.

-r *remotehost*
Process automatic login from *remotehost*. Used by *rlogind* to allow a user with the proper permissions to bypass the password prompt when logging in.

See Also

login(TC), rlogind(ADMN), telnetd(ADMN), rhosts(SFF).

ping

send ICMP ECHO_REQUEST packets to network hosts

Syntax

/etc/ping [**-r**] [**-v**] host [packetsize] [count]

Description

ping is a troubleshooting tool for tracking a single-point hardware or software failure in the Internet. It uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (*pings*) have an IP and an ICMP header, followed by a **struct timeval** and an arbitrary number of pad bytes used to fill out the packet. Default datagram length is 64 bytes, but this may be changed using the command-line option. Other options are:

-r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it.

-v Verbose output. ICMP packets other than ECHO RESPONSE that are received are listed.

When using *ping* for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be pinged. The *ping* tool sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. No output is produced if there is no response. If an optional *count* is given, only that number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times are out (with a *count* specified), or if the program is terminated with a SIGINT, then a brief summary is displayed.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use *ping* during normal operations or from automated scripts.

See Also

netstat(TC), ifconfig(ADMN).

rdate

notify time server that date has changed

Syntax

rdate

Description

rdate notifies *timed*(ADMN) that the system date has changed. If the local time server is a master, it will notify all of the slaves that the time has changed. If it is a slave, it will ask the master to update the time.

rdate should be run whenever the super user sets the date with *date(C)*. A shell script such as the following could be used to do both automatically.

```
:
: mv /bin/date /bin/s5date
: install as /bin/date
:
PATH=/bin:/usr/bin
s5date $*
rdate
```

See Also

date(C), adjtime(SSC), gettimeofday(SLIB), icmp(ADMP), timed(ADMN), timedc(ADMN).

rexecd

remote execution server

Syntax

/etc/rexecd

Description

rexecd is the server for the *rexec* (SLIB) routine. The server provides remote execution facilities with authentication based on user names and passwords.

rexecd listens for service requests at the port indicated in the *exec* service specification; see *services* (SFF). When a service request is received, the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null-terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null-terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.

5) A null-terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) Then, *rexecd* validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail, the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by *rexecd*.

rexecd is started by the super-server *inetd*, and therefore must have an entry in *inetd*' s configuration file **/etc/inetd.conf**.

Diagnostics

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1. (0 is returned in step 7, above, upon successful completion of all the steps prior to the command execution.)

"username too long"
The name is longer than 16 characters.

"password too long"
The password is longer than 16 characters.

"command too long "
The command line passed exceeds the size of the argument list (as configured into the system).

"Login incorrect."
No password file entry for the user name existed.

"Password incorrect."
The wrong password was supplied.

"No remote directory."
The *chdir* command to the home directory failed.

"Try again."
A *fork* by the server failed.

"<shellname>: ..."
The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

See Also

rexec(SLIB), inetd(ADMN), inetd.conf(SFF), services(SFF).

Notes

Indicating "Login incorrect'' as opposed to "Password incorrect'' is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.

rlogind

remote login server

Syntax

/etc/rlogind

Description

rlogind is a network server that supports remote logins by programs such as *rlogin*(TC). It is started by the superserver *inetd* and, therefore, must have an entry in *inetd*'s configuration file **/etc/inetd.conf.** [See *inetd*(ADMN) and *inetd.conf*(SFF).]

rlogind enforces an authentication procedure based on equivalence of user names (see *rhosts*(SFF)). This procedure assumes all hosts on the network are equally secure.

See Also

inetd(ADMN), rlogin(TC), inetd.conf(SFF), rhosts(SFF), services(SFF).

rmail

handle remote mail received via uucp

Syntax

rmail user ...

Description

rmail interprets incoming mail received via *uucp*(C), collapsing "From" lines in the form generated by *mail*(TC) into a single line of the form *return-path!sender*, and passing the processed mail on to *sendmail*(ADMN).

rmail is explicitly designed for use with *uucp* and *sendmail*.

See Also

mail(TC), uucp(C), sendmail(ADMN).

route

manually manipulate the routing tables

Syntax

/etc/route [**-f**] [**-n**] [command destination gateway [metric]]

Description

route is a program used to manipulate manually the network routing tables. It is normally not needed, since the routing daemon *routed* manages the system routing table and therefore handles this function.

route accepts two commands: *add*, to add a route; and *delete*, to delete a route.

All commands have the following syntax:

/etc/route *command destination gateway* [*metric*]

where *destination* is a host or network for which the route is "to", *gateway* is the gateway to which packets should be addressed, and *metric* is an optional count indicating the number of hops to the *destination*. If no metric is specified, *route* assumes a value of 0. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. If the *destination* has a local address part of INADDR_ANY, the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. Note: If the route is to a destination connected via a gateway, *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first in the host-name database; see *hosts(SFF)*. If this lookup fails, the name is then looked for in the network name database; see *networks(SFF)*.

route uses a raw socket and the SIOCADDRT and SIOCDELRT *ioctl*'s to do its work. Therefore, only the super user may modify the routing tables.

If the **-f** option is specified, *route* will flush the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command's application.

The **-n** option prevents attempts to print host and network names symbolically when reporting actions.

Diagnostics

add [host | network]
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl* call.

"delete *host*: **gateway** *host* **flags** *hex-flags***"**
As above, but when deleting an entry.

"*host host* **done"**
When the **-f** flag is specified, each routing table entry deleted is indicated with a message of this form.

"not in table"
A delete operation was attempted for an entry which was not present in the tables.

"routing table overflow"
An add operation was attempted, but the system was low on resources and unable to allocate memory to create the new entry.

See Also

routed(ADMN), intro(ADMN), hosts(SFF), networks(SFF).

routed

network routing daemon

Syntax

/etc/routed [**-d**] [**-g**] [**-s**] [**-t**] [logfile]

Description

routed manages the Internet routing tables using a variant of the Xerox NS Routing Information Protocol. *routed* is invoked by the superuser (usually during init 2).

In normal operation, *routed* listens on the *udp(ADMP)* socket for the *route* service (see *services(SFF))* for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When *routed* is started, it uses the SIOCGIFCONF *ioctl* to find those directly connected interfaces configured into the system and marked "up". (The software loopback interface is ignored.) If multiple interfaces are present, it is assumed that the host will forward packets between networks. Then, *routed* transmits a request packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for request and response packets from other hosts.

When a request packet is received, *routed* formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a *hop count* metric. (A count of 16 or greater is considered infinite.) The metric associated with each route returned provides a metric relative to the sender.

Response packets received by *routed* are used to update the routing tables if one of the following conditions is satisfied:

(1)
No routing table entry exists for the destination network or host, and the metric indicates the destination is reachable (that is, the hop count is not infinite).

(2)
The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

(3)
The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

(4)
The new route describes a shorter path to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, *routed* records the change in its internal tables and updates the kernel-routing table. The change is reflected in the next response packet sent.

In addition to processing incoming packets, *routed* also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, its metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to ensure that the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly-connected hosts and networks. The response is sent to the broadcast address on nets capable of the broadcast function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

routed supports several options:

-d Enable additional debugging information to be logged, such as bad packets received.

-g This flag is used on internetwork routers to offer a route to the default destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

-s Supplying this option forces *routed* to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

-q This is the opposite of the **-s** option.

-t If the **-t** option is specified, all packets sent or received are printed on the standard output. In addition, *routed* will not divorce itself from the controlling terminal, and so interrupts from the keyboard will kill the process.

Any other argument supplied is interpreted as the name of file in which *routed*'s actions should be logged. This log contains information about any changes to the routing tables and, if the log is not tracing all packets, a history of recent messages sent and received that are related to the changed route.

In addition to the facilities described above, *routed* supports the notion of distant passive and active gateways. When *routed* is started up, it reads the file **/etc/gateways** to find gateways that may not be located using only information from the SIOCGIFCONF *ioctl*. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a *routed* process running on the machine). Passive gateways are maintained in the routing tables forever, and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally with network interfaces. Routing information is distributed to the gateway and, if no routing information is received for a period of time, the associated route is deleted. External gateways are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to inform *routed* that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The **/etc/gateways** is comprised of a series of lines, each in the following format:

<net | host> *name1* **gateway** *name2* **metric** *value* **<passive | active | external>**

The **net** or **host** keyword indicates whether the route is to a network or specific host.

name1 is the name of the destination network or host. This may be a symbolic name located in **/etc/networks** or **/etc/hosts** (or, if started after *named(ADMN)*, known to the name server), or an Internet address specified in "dot" notation; see *hosts(SFF)* and *inet(ADMP)*.

name2 is the name or address of the gateway to which messages should be forwarded.

value is a metric indicating the hop count to the destination host or network.

One of the keywords **passive**, **active** and **external** indicates whether the gateway should be treated as passive or active (as described above), or the gateway is external to the scope of the *routed* protocol.

Files

/etc/gateways	for distant gateways

See Also

udp(ADMP).

Notes

The kernel's ICMP routing tables may not correspond to those of *routed* when ICMP redirects change or add routes.

rshd

remote shell server

Syntax

/etc/rshd

Description

rshd is the network server for programs such as *rcmd*(TC) and *rcp*(TC) which need to execute a noninteractive shell on remote machines. *rshd* is started by the superserver *inetd*, and therefore must have an entry in *inetd*'s configuration file **/etc/inetd.conf**. [See *inetd*(ADMN) and *inetd.conf*(SFF)].

rshd enforces an authentication procedure based on equivalence of user names (see *rhosts*(SFF)). This procedure assumes all nodes on the network are equally secure.

See Also

inetd(ADMN), rcmd(TC), rcp(TC), inetd.conf(SFF), rhosts(SFF).

rwhod

system status server

Syntax

/etc/rwhod

Description

rwhod is the server which maintains the database used by the *rwho*(TC) and *ruptime*(TC) programs. Its operation is predicated on the ability to broadcast messages on a network.

rwhod operates as both a producer and a consumer of status information. As a producer of information, it periodically queries the state of the system and constructs status messages that are broadcast on a network. As a consumer of information, it listens for other *rwhod* servers' status messages, validating them, then recording them in a collection of files located in the directory **/usr/spool/rwho.**

The server transmits and receives messages at the port indicated in the *rwho* service specification; see *services*(SFF). The messages sent and received are of the form:

```
struct  outmp {
        char    out_line[8];/* tty name */
        char    out_name[8];/* user id */
        long    out_time;/* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the *uptime*(C) program, and represent load averages over the 5-, 10-, and 15- minute intervals prior to a server's transmission; they are multiplied by 100 for representation in an integer. The host name included is that returned by the *gethostname*(SLIB) system call, with any trailing domain name omitted. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the *utmp*(M) entry for each non-idle terminal line and a value indicating the time in seconds since a character was last received on the terminal line.

Messages received by the *rwho* server are discarded unless they originated at an *rwho* server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by *rwhod* are placed in files named **whod.hostname** in the directory **/usr/spool/rwho**. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 5 minutes. *rwhod* performs an *nlist*(S) on **/unix** every 30 minutes to guard against the possibility that this file is not the system image currently operating.

See Also

rwho(TC), ruptime(TC).

Notes

There should be a way to relay status information between networks. Status information should be sent only upon request, rather than continuously. People often interpret the server dying or network communication failures as a machine going down.

Some mechanism for cleaning dead machine data out of the spool directory is needed.

sendmail

send mail over the internet

Syntax

/usr/lib/sendmail [flags] [address ...]

newaliases

mailq [**-v**]

Description

sendmail sends a message to one or more recipients, routing the message over whatever networks are necessary. *sendmail* does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; *sendmail* is used only to deliver preformatted messages.

With no flags, *sendmail* reads its standard input up to an end-of-file or a line consisting only of a single dot and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use, based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally, the sender is not included in any alias expansions; for instance, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

Flags are:

-ba	Go into ARPANET mode. Every input line must end with a CR-LF, and each message will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.
-bd	Run as a daemon. *sendmail* will fork and run in background listening on TCP port 25 for incoming SMTP connections. This is normally run from */etc/rc*.
-bi	Initialize the alias database. This works only if *sendmail* was built with a DBM library. Otherwise, this option does nothing.

-bm	Deliver mail in the usual way (default).
-bp	Print a listing of the queue.
-bs	Use the SMTP protocol as described in RFC821 on standard input and output. This flag implies all the operations of the **-ba** flag that are compatible with SMTP.
-bt	Run in address-test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.
-bv	Verify names only; do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.
-bz	Create the configuration freeze file.
-C*file*	Use alternate configuration file. *sendmail* refuses to run as root if an alternate configuration file is specified. The frozen configuration file is bypassed.
-d*X*	Set debugging value to *X*.
-F*fullname*	Set the full name of the sender.
-f*name*	Sets the name of the "from" person (that is, the sender of the mail). **-f** can only be used by trusted users (normally root, daemon, and network), or if the person you are trying to become is the same as the person you are.
-h*N*	Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, "Received:'' lines in the message are counted.
-n	Don't do aliasing.
-o*x value*	Set option *x* to the specified *value*. Options are described below.

-q[*time*]	Process saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with 's' being seconds, 'm' being minutes, 'h' being hours, 'd' being days, and 'w' being weeks. For example, "-q1h30m" or "-q90m" would both set the timeout to one hour and thirty minutes. If *time* is specified, *sendmail* will run in background. This option can be used safely with **-bd**.
-r*name*	An alternate and obsolete form of the **-f** flag.
-t	Read message for recipients. To:, Cc:, and Bcc: lines will be scanned for recipient addresses. The Bcc: line will be deleted before transmission. Any addresses in the argument list will be suppressed, that is, they will *not* receive copies even if listed in the message header.
-v	Go into verbose mode. Alias expansions will be announced, and so on.

There is also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **-o** flag or in the configuration file. These are described in detail in the *TCP/IP Administrator's Guide*. The options are:

A*file*	Use alternate alias file.
c	On mailers that are considered expensive to connect to, do not initiate immediate connection. This requires queueing.
d*x*	Set the delivery mode to *x*. Delivery modes are 'i' for interactive (synchronous) delivery, 'b' for background (asynchronous) delivery, and 'q' for queue only - that is, actual delivery is done the next time the queue is run.
D	Try to rebuild the alias database automatically if necessary.

e*x*	Set error processing to mode *x*. Valid modes are 'm' to mail back the error message, 'w' to "write" back the error message (or mail it back if the sender is not logged in), 'p' to print the errors on the terminal (default), 'q' to throw away error messages (so that only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by mode 'm' or 'w' and if the sender is local to this machine, a copy of the message is appended to the file **dead.letter** in the sender's home directory.
F*mode*	The mode to use when creating temporary files.
f	Save UNIX-style From lines at the front of messages.
g*N*	The default group id to use when calling mailers.
H*file*	The SMTP help file.
i	Do not take dots on a line by themselves as a message terminator.
m	Send to "me" (the sender) also if I am in an alias expansion.
o	If set, this message may have old-style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.
Q*queuedir*	Select the directory in which to queue messages.
r*timeout*	The timeout on reads; if none is set, *sendmail* will wait forever for a mailer. This option violates the word (if not the intent) of the SMTP specification, so the timeout should probably be fairly large.
S*file*	Save statistics in the named file.
s	Always instantiate the queue file, even under circumstances where it is not strictly necessary. This provides safety against system crashes during delivery.
T*time*	Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (for instance, because a host is down) for this amount of time, failed messages will be returned to the sender. The default is three days.

t*stz,dtz*	Set the name of the time zone.
u*N*	Set the default user id for mailers.

In aliases, the first character of a name may be a vertical bar to cause interpretation of the rest of the name as a command to which to pipe the mail. It may be necessary to quote the name to keep *sendmail* from suppressing the blanks between arguments. For example, a common alias is:

```
msgs: "|/usr/ucb/msgs -s"
```

Aliases may also have the syntax ":include:*filename*" to ask *sendmail* to read the named file for a list of recipients. For example, an alias such as:

```
poets: ":include:/usr/local/lib/poets.list"
```

would read **/usr/local/lib/poets.list** for the list of addresses making up the group.

The *sendmail* command returns an exit status describing what it did. The codes are defined in <*sysexits.h*>:

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall, meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating-system error, such as cannot fork.
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as *newaliases, sendmail* will rebuild the alias database. This works only if *sendmail* was built with a DBM library. Otherwise, this option does nothing. If invoked as *mailq, sendmail* will print the contents of the mail queue.

Files

Except for **/usr/lib/sendmail.cf**, these pathnames are all specified in **/usr/lib/sendmail.cf**. Thus, these values are only approximations.

/usr/lib/aliases	raw data for alias names
/usr/lib/sendmail.cf	configuration file
/usr/lib/sendmail.fc	frozen configuration
/usr/lib/sendmail.hf	help file

/usr/lib/sendmail.st	collected statistics
/usr/spool/mqueue/*	temp files

See Also

mail(TC), aliases(SFF), mailaddr(SFF);
RFC819, RFC821, RFC822;
The chapter "Introduction to sendmail" in the *TCP/IP Administrator's Guide*;
The chapter "Installing and Operating Sendmail" in the *TCP/IP Administrator's Guide*.

slattach, sldetach

attach and detach serial lines as network interfaces

Syntax

/etc/slattach devname source destination [baudrate]

/etc/sldetach interface-name

Description

slattach is used to assign a serial (tty) line to a network interface using the DARPA Internet Protocol, and to define the source and destination network addresses. The *devname* parameter is the name of the device the serial line is attached to, that is, /dev/tty001. The source and destination are either host names present in the host name data base (see *hosts(SFF)),* or DARPA Internet addresses expressed in the Internet standard "dot notation." The optional *baudrate* parameter is used to set the speed of the connection; if not specified, the default of 9600 is used.

Only the superuser may attach or detach a network interface.

There should not be a *getty* (M) on the line.

sldetach is used to remove the serial line that is being used for IP from the network tables and allow it to be used as a normal terminal again. *interface-name* is the name that is shown by *netstat* (TC).

Examples

```
/etc/slattach tty001 tom-src genstar
/etc/slattach /dev/tty001 hugo dahl 4800
/etc/sldetach sl01
```

Files

/etc/hosts
/dev/*
/usr/spool/locks/slippid.*

Diagnostics

Various messages indicating:
- the specified interface does not exist
- the requested address is unknown
- the user is not the superuser

See Also

hosts(SFF), netstat(TC), ifconfig(ADMN).

slink

streams linker

Syntax

slink [-v] [-f] [**-c** file] [func [arg1 arg2 ...]]

Description

slink is a STREAMS configuration utility that is used to link together the various STREAMS modules and drivers required for STREAMS TCP/IP. Input to *slink* is in the form of a script specifying the STREAMS operations to be performed. Input is normally taken from the file **/etc/strcf**.

The following options may be specified on the *slink* command line:

-c *file*	Use *file* instead of **/etc/strcf**.
-v	Verbose mode (that is, each operation is logged to **stderr**).
-f	Do not fork (that is, *slink* will remain in foreground).

The configuration file contains a list of functions, each of which is composed of a list of commands. Each command is a call to one of the functions defined in the configuration file or to one of a set of built-in functions. Among the built-in functions are the basic STREAMS operations *open*, *link*, and *push*, along with several TCP/IP-specific functions.

slink processing consists of parsing the input file, then calling the user-defined function **boot**, which is normally used to set up the standard configuration at boot time. If a function is specified on the *slink* command line, that function will be called instead of **boot**. Following the execution of the specified function, *slink* goes into the background and remains idle, holding open whatever file descriptors have been opened by the configuration commands.

A function definition has the following form:

```
function-name {
            command1
            command2
            ...
}
```

The syntax for commands is:

```
function arg1 arg2 arg3 ...
```

or:

```
var = function arg1 arg2 arg3 ...
```

The placement of newlines is important: a newline must follow the left and right braces and every command. Extra newlines are allowed, that is, where one newline is required, more than one may be used. A backslash ('\') followed immediately by a newline is considered equivalent to a space, so it may be used to continue a command on a new line. The use of other white space characters (spaces and tabs) is at the discretion of the user, except that there must be white space separating the function name and the arguments of a command.

Comments are delimited by '#' and newline, and are considered equivalent to a newline.

Function and variable names may be any string of characters taken from A-Z, a-z, 0-9, and '_', except that the first character cannot be a digit. Function names and variable names occupy separate name spaces. All functions are global and may be forward-referenced. All variables are local to the functions in which they occur.

Variables are defined when they appear to the left of an equal sign ('=') on a command line, such as:

```
tcp = open /dev/inet/tcp
```

The variable acquires the value returned by the command. In the above example, the value of the variable *tcp* will be the file descriptor returned by the *open* call.

Arguments to a command may be variables, parameters, or strings.

A variable that appears as an argument must have been assigned a value on a previous command line in that function.

Parameters take the form of a dollar sign ('$') followed by one or two decimal digits, and are replaced with the corresponding argument from the function call. If a given parameter was not specified in the function call, an error results (for instance, if a command references $3 and only two arguments were passed to the function, an execution error will occur).

Strings are sequences of characters optionally enclosed in double quotes ('"'). Quotes may be used to prevent a string from being interpreted as a variable name or a parameter, and to allow the inclusion of spaces, tabs, and the special characters '{', '}', '=', and '#'. The backslash ('\') may also be used to quote the characters '{', '}', '=', '#', '"', and '\' individually.

The following built-in functions are provided by *slink*:

open *path*
: Open the device specified by pathname *path*. Returns a file descriptor referencing the open stream.

link *fd1 fd2*
: Link the stream referenced by *fd2* beneath the stream referenced by *fd1*. Returns the link identifier associated with the link. *Note*: The *fd2* function cannot be used after this operation.

push *fd module*
: Push the module identified by *module* onto the stream referenced by *fd*.

sifname *fd link name*
: Send a SIOCSIFNAME (set interface name) ioctl down the stream referenced by *fd* for the link associated with link identifier *link* specifying the name given in *name*.

unitsel *fd unit*
: Send a IF_UNITSEL (unit select) ioctl down the stream referenced by *fd* specifying the unit given in *unit*.

dlattach *fd unit*
: Send a DL_ATTACH_REQ message down the stream referenced by *fd* specifying the unit given in *unit*.

initqp *path qname lowat hiwat ...*
: Send an INITQPARMS (initialize queue parameters) ioctl to the driver corresponding to pathname *path*. *qname* specifies the queue for which the low and high water marks will be set, and must be one of:

hd	stream head
rq	read queue
wq	write queue
muxrq	multiplexor read queue
muxwq	multiplexor write queue

The *lowat* and *hiwat* functions specify the new low and high water marks for the queue. Both *lowat* and *hiwat* must be present. To change only one of these parameters, the other may be replaced with a dash ('-'). Up to five *qname lowat hiwat* triplets may be present.

strcat *str1 str2*
: Concatenate strings *str1* and *str2* and return the resulting string.

return *val*	Set the return value for the current function to *val*. *Note*: executing a **return** command does not terminate execution of the current function.

Files

/etc/strcf

See Also

strcf(SFF), intro(ADMP).

talkd

remote user communication server

Syntax

/etc/talkd

Description

Talkd is the server that notifies a user that somebody else wants to initiate a conversation. It acts as a repository of invitations, responding to requests by clients wishing to rendezvous to hold a conversation. In normal operation, a *talk* client initiates a rendezvous by sending a CTL_MSG to the server of type LOOK_UP (see <*protocols/talkd.h*>). This causes the server to search its invitation tables to check if an invitation currently exists for the client. If the lookup fails, the caller then sends an ANNOUNCE message causing the server to broadcast an announcement on the callee's login ports requesting contact. When the callee responds, the local server uses the recorded invitation to respond with the appropriate rendezvous address and the caller and callee client programs establish a stream connection through which the conversation takes place.

See Also

talk(TC), write(TC)

/etc/tcp

TCP start/stop script

Syntax

/etc/tcp start
/etc/tcp stop

Description

/etc/tcp is used to start or stop the STREAMS TCP software. TCP will start automatically at system startup time if **/etc/rc.d/6/***name* contains a script including the command **/etc/tcp start**. TCP does not stop automatically at system shutdown time. The command **/etc/tcp stop** will stop TCP. See *init*(M) for further information.

/etc/tcp must be customized for a particular installation before it can be used. The following items must be edited:

Domain name	The environment variable *DOMAIN* must be set to the name of your domain.
Interface configuration	*ifconfig* commands must be used to set the internet address (and any other desired options) for each of your interfaces. The *ifconfig* line for the loopback interface should not require modification. See *ifconfig*(ADMN) for further information.

The following items may need to be edited:

PATH	The supplied path may require modification if commands run by */etc/tcp* are in other directories.
PROCS	The *PROCS* variable contains a space-separated list of names of processes to kill when executing the **stop** function. If additional daemons are used, their names can be added to this list.
Network initialization	Certain network hardware may require the execution of an initialization command before use. Any such commands should be included in this section.

Daemons

The standard internetworking daemons are started at this point. Any additional daemons or other commands may be included in this section. Any of the standard daemons that are not desired may be removed or commented out.

telnetd

DARPA TELNET protocol server

Syntax

/etc/telnetd

Description

telnetd is a server that supports the DARPA standard **TELNET** virtual terminal protocol. *telnetd* is invoked by the internet server (see *inetd*(ADMN)), normally for requests to connect to the **TELNET** port as indicated by the **/etc/services** file (see *services*(SFF)).

telnetd operates by allocating a pseudo-terminal device for a client, then creating a login process that has the slave side of the pseudo terminal as **stdin**, **stdout**, and **stderr**. *telnetd* manipulates the master side of the pseudo-terminal, implementing the **TELNET** protocol and passing characters between the remote client and the login process.

When a **TELNET** session is started up, *telnetd* sends **TELNET** options to the client side indicating a willingness to do remote echo of characters, to suppress go ahead, and to receive terminal type information from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in ICANON mode, and with TAB3 and ICRNL enabled. (See *termio*(M).)

telnetd is willing to do: *echo*, *binary*, *suppress go ahead*, and *timing mark*. *telnetd* is willing to have the remote client do: *binary*, *terminal type*, and *suppress go ahead*.

See Also

telnet(TC)

Notes

Some **TELNET** commands are only partially implemented.

The **TELNET** protocol allows for the exchange of the number of lines and columns on the user's terminal, but *telnetd* does not make use of them.

Because of bugs in the original 4.2 BSD *telnet, telnetd* performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD *telnet.*

Binary mode has no common interpretation except between similar operating systems (Unix, in this case).

The terminal type name received from the remote client is converted to lowercase.

The packet interface to the pseudo terminal should be implemented for intelligent flushing of input and output queues.

telnetd never sends **TELNET** *go ahead* commands.

tftpd

DARPA Trivial File Transfer Protocol server

Syntax

/etc/tftpd

Description

tftpd is a server that supports the DARPA Trivial File Transfer Protocol. The TFTP server operates at the port indicated in the *tftp* service description; see *services* (SFF). This port number may be overridden (for debugging purposes) by specifying a port number on the command line.

The use of *tftp* does not require an account or password on the remote system. Due to the lack of authentication information, *tftpd* will allow only publicly readable files to be accessed. Note that this extends the concept of public to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling tftp service.

tftpd is spawned by the superserver *inetd* and, therefore, must have an entry in *inetd*'s configuration file, **/etc/inetd.conf**. [See *inetd* (ADMN) and *inetd.conf* (SFF).] Note that the *tftpd* entry in this file must be "wait": this is to prevent subsequent *selects* from being successful before the first *tftpd* process does its *receive*. *tftpd* takes care to prevent multiple *tftpd* processes from being spawned to service the same request. (*inetd* is able to continue processing new messages on the port.)

See Also

inetd(ADMN), tftp(TC), inetd.conf(SFF), services(SFF).

Warnings

This server is known only to be self-consistent (that is, it operates with the user TFTP program *tftp* (TC)).

The search permissions of the directories leading to the files accessed are not checked if *tftp* runs as root. The default configuration runs *tftpd* as user "sync."

timed

time server daemon

Syntax

/etc/timed [-t] [-M] [-n network] [-i network]

Description

timed is the time server daemon and is normally invoked at boot time from the STREAMS TCP/IP start-up script. It synchronizes the host's time with that of other machines in a local area network running *timed*(ADMN). These time servers will slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. The average network time is computed from measurements of clock differences using the ICMP timestamp request message.

The service provided by *timed* is based on a master-slave scheme. When *timed*(ADMN) is started on a machine, it asks the master for the network time and sets the host's clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls *adjtime*(SSC) to perform the needed corrections on the host's clock.

It also communicates with *rdate*(ADMN) in order to set the date globally, and with *timedc*(ADMN), a timed control program. If the machine running the master crashes, then the slaves will elect a new master from among slaves running with the **-M** flag. A *timed* running without the **-M** flag will remain a slave. The **-t** flag enables *timed* to trace the messages it receives in the file **/usr/adm/timed.log**. Tracing can be turned on or off by the program *timedc(ADMN)*. *timed* normally checks for a master time server on each network to which it is connected, except as modified by the options described below. It will request synchronization service from the first master server located. If permitted by the **-M** flag, it will provide synchronization service on any attached networks on which no current master server was detected. Such a server propagates the time computed by the top-level master. The **-n** flag, followed by the name of a network to which the host is connected (see *networks(SFF)),* overrides the default choice of the network addresses made by the program. Each time the **-n** flag appears, that network name is added to a list of valid networks. All other networks are ignored. The **-i** flag, followed by the name of a network to which the host is connected (see *networks(SFF)),* overrides the default choice of the network addresses made by the program. Each time the **-i** flag appears, that network name is added to a list of networks to ignore. All other networks are used by the time daemon. The **-n** and **-i** flags are meaningless if used together.

Files

/usr/adm/timed.log	tracing file for timed
/usr/adm/timed.masterlog	log file for master timed

See Also

date(C), adjtime(SSC), gettimeofday(SLIB), icmp(ADMP), rdate(ADMN), timedc(ADMN).

timedc

timed control program

Syntax

timedc [command [argument ...]]

Description

timedc is used to control the operation of the *timed* program. It may be used to:

- measure the differences between machines' clocks,
- find the location where the master time server is running,
- enable or disable tracing of messages received by *timed*, and
- perform various debugging actions.

Without any arguments, *timedc* will prompt for commands from the standard input. If arguments are supplied, *timedc* interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected, causing *timedc* to read commands from a file. Commands may be abbreviated; recognized commands are:

? [*command* ...]

help [*command* ...]
Print a short description of each command specified in the argument list or, if no arguments are given, a list of the recognized commands.

clockdiff *host* ...
Compute the differences between the clock of the host machine and the clocks of the machines given as arguments.

trace { *on* | *off* }
Enable or disable the tracing of incoming messages to *timed* in the file **/usr/adm/timed.log**.

quit
Exit from timedc.

Other commands may be included for use in testing and debugging *timed*; the help command and the program source may be consulted for details.

Files

/usr/adm/timed.log	tracing file for timed
/usr/adm/timed.masterlog	log file for master timed

See Also

date(C), adjtime(SSC), icmp(ADMP), rdate(ADMN), timed(ADMN).

Diagnostics

?Ambiguous command	abbreviation matches more than one command
?Invalid command	no match found
?Privileged command	command can be executed by root only

trace, query

routing tools

Syntax

trace [on|off] machines... **query** [-n] hosts...

Description

trace sends a RIP_TRACE_ON or RIP_TRACE_OFF command to the specified machines. *Machine must be specified as an IP address.*

query is used to request routing information from the specified host. Any packets received in response to a query will be displayed.

These commands are useful for debugging *routed(ADMN).*

See Also

routed(ADMN), udp(ADMP).
RFC1058

Bugs

RFC 1058 states that TRACE_ON and TRACE_OFF are not supposed to be supported any more.

trpt

transliterate protocol trace

Syntax

trpt [**-a**] [**-s**] [**-t**] [**-f**] [**-j**] [**-p** hex-address] [system [core]]

Description

trpt interrogates the buffer of TCP trace records created when a socket is marked for debugging (see *getsockopt*(SSC)), and prints a readable description of these records. When no options are supplied, *trpt* prints all the trace records found in the system, grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

-a In addition to the normal output, print the values of the source and destination addresses for each packet recorded.

-s In addition to the normal output, print a detailed description of the packet sequencing information.

-t In addition to the normal output, print the values for all timers at each point in the trace.

-f Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

-j Just give a list of the protocol control block addresses for which there are trace records.

-p Show only trace records associated with the protocol control block, the address of which follows.

The recommended use of *trpt* is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **-A** option to *netstat*(TC). Then run *trpt* with the **-p** option, supplying the associated protocol control block addresses. The **-f** option can be used to follow the trace log, once the trace is located. If there are many sockets using the debugging option, the **-j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

Files

/unix
/dev/kmem

See Also

getsockopt(SSC), netstat(TC)

Diagnostics

The message "no namelist'' when the system image doesn't contain the proper symbols to find the trace buffer; other messages which should be self explanatory.

Bugs

Should also print the data for each input or output, but this is not saved in the trace record.

The output format is inscrutable and should be described here.